タブローの最適配置問題 (計算機科学とアルゴリズムの数理的基礎とその応用)

AUTHOR(S):

安齋, 進也; 全, 眞嬉; 葛西, 亮生; コルマン, マティアス; 徳山, 豪

CITATION:

安齋, 進也 ...[et al]. タブローの最適配置問題 (計算機科学とアルゴリズムの数理的基礎とその応用). 数理解析研究所講究録 2011, 1744: 93-98

ISSUE DATE:

2011-06

URL:

http://hdl.handle.net/2433/170967

数理解析研究所講究録
第1744巻 2011年 93-98

タブローの最適配置問題

安齋 進也* 全 眞嬉* 葛西 亮生* コルマン マティアス† 徳山 豪*

1 はじめに

情報処理技術が発達している中で, 画像処理技術も進歩し, 広く実用化がなされている. これは, 画像が視覚的に情報を表現することができるメディアであり, 文字や音声に比べ直感的でわかりやすいといった特徴を持っているからである. 画像処理技術には, コンピュータを用いて与えられた画像に含まれている物体の認識やその画像が持っている特徴の解析, さらには, 画像を見やすくするための変換など様々な技術が挙げられる. これらの中でも画像切り出しと呼ばれる問題は, 重要な問題の1つとして古くから研究がなされてきた.

画像切り出しとは, 与えられた画像からイメージを切り出す作業である (図1.1). この操作は, パターン認識や特徴抽出等を行うときに用いられるため, 様々な手法が提案されてきた. この問題に対し, Asano ら [1] は組合せ最適化問題として定式化し, パラメトリック最適化の技法を利用して解決する枠組みを提案した.

画像切り出し問題の入力として, ピクセル画像を表す $n \times n$ のピクセル平面 $\mathbf{P}$ を考える. ピクセル平面 $\mathbf{P}$ の各ピクセル p には重み $b(p)$ が与えられている. 画像切り出しにより得られる画像は良い性質を持つピクセル集合であるから, ピクセル画像から良いピクセル集合を選べば良い. このことから, 画像切り出し問題を組合せ最適化問題として考えることができる.

これまでの研究では, ある基準を用いて最適な切り出しを行うのではなく, 画像の性質に応じて発見的に切り出す画像を求めていた. Asano ら [1] は組合せ最適化問題として定式化することにより, 切り出された画像の品質に理論的保証を与える手法を提案した. 具体的には, ピクセル集合の族 $\mathcal{F}$ と, $\mathcal{F}$ 上の

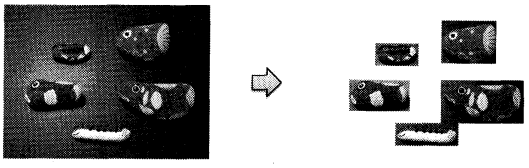

図 1.1. 画像切り出し

凸性を持つ実数関数 Φ を定義し, $\Phi(R)$ を最大化するピクセル集合 $R \in \mathcal{F}$ を求めるという手法である. この定式化の中で重要となる問題が最大重み領域問題と呼ばれる問題である.

次に最大重み領域問題について述べる. $n \times n$ のピクセル平面 $\mathbf{P}$ と, ピクセル領域と呼ばれるピクセル集合の族 $\mathcal{F} \subseteq 2^{\mathbf{P}}$ を考える. ピクセルとは, 単位正方形 $p(i,j) = [i-1,i] \times [j-1,j]$ のことである. ここで, $1 \leq i,j \leq n$ とする. ピクセル座標 (i,j) に位置するピクセルは実数値 $w(p) = w(i,j)$ を持ち, これを p の重みと呼ぶ. 重みには負の値も存在する. 従って, ピクセル平面 $\mathbf{P}$ に対する重みを表す行列 $W = (w_{i,j})$ は実数値行列で表される. 便宜上, $1 \leq i,j \leq n$ に対して, $w(0,j) = w(n+1,j) = w(i,0) = w(i,n+1) = 0$ とする. 最大重み領域問題とは次のように表される.

最大重み領域問題

入力 $n \times n$ の重み行列 W
出力 $W(R) = \sum_{p \in R} w(p)$ を最大化する領域 $R \in \mathcal{F}$

この問題の難しさはピクセル集合の領域族 $\mathcal{F}$ に依存する. $\mathcal{F} = 2^{\mathbf{P}}$ の場合, この問題の解は自明であり, 最適なピクセル領域は重みが正であるピクセルの集合として表される. 一方で, $\mathcal{F}$ が $\mathbf{P}$ 内において, 4近傍連結であるような領域全体のとき, この問題は NP 困難であることが Asano ら [1] により示されている. このように, 最大重み領域問題の難しさは領域族 $\mathcal{F}$ に依存する. しかしながら, ピクセル平面 $\mathbf{P}$ のサイ

*東北大学大学院情報科学研究科
†ブリュッセル自由大学

ズ $n \times n = N$ に対して, 多項式で解ける領域族も多く存在する. さらに, Chun ら [3] はより一般的な領域族を考え, それらの領域族に対する効率的なアルゴリズムを与えた. 具体的には, ピクセル平面 $\mathbf{P}$ のサイズ N に対して多項式時間で解けることが既にわかっている領域を基本図形とする. そして, それらの図形の非交差和領域を領域族とした最大重み領域問題を $N(= n \times n)$ に関する多項式時間で解けることを示した. 最大重み領域問題は前述の通り, 画像処理分野において重要な問題であるが, 他にもデータマイニングや放射線医療の最適化にも応用されている.

本論文では, 基本図形として長方形とタブローと呼ばれる図形を考え, この図形の非交差和領域を領域族とした最大重み領域問題を解くアルゴリズムを考える. タブローとは, ある点 p を同じ隅に持つ長方形の和集合で表される領域のことである. これ以降では, この問題を各ピクセルが重みを持つ平面に重みの総和が最大となる長方形やタブローを配置する問題とみなして考えていく. また, 最適化の基準として, 重みの総和を最大化する場合に加えて, 最小値を最大化する場合についても考える.

1.1 既存研究

ここでは, 長方形の最適配置問題に対する既存研究について述べる. 長方形配置の総和最大化問題に対して, Bae と Takaoka[2] により, 貪欲算法を用いて k 個の長方形を配置する手法が提案された. この手法は, 重み行列から和が最大となる部分行列を求め, 選択された重みを $-\infty$ に変換する. そのようにして得られた重み行列に対し, 同じ操作を繰り返すことにより k 個の長方形を求めている. この操作を効率的に繰り返すため, ノードが部分行列の範囲を表し, 根が最適な範囲を持つような木を作成する. 木の作成に $O(n^3)$ 時間かかり, 重みが $-\infty$ に変換された後の木の再構築に $O(n^2 \log n)$ かかるため, 全体の時間計算量は $O(n^3 + kn^2 \log n)$ となる. しかし, この手法は必ずしも最適解を出力するとは限らない. 実際に, この問題が NP 困難であることは, Korman[5] により示されている. また, 重みの最小値を最大化する問題の計算の難しさについては特に証明はされていないが, 地図上へのラベル配置問題との類似性から, NP 困難であることが予想される. ラベル配置問題は平面上に与えられた点にそれぞれのラベル同士が重ならないように, かつ障害物と重ならないようにラベルを配置する問題である. さらに, ラベルをすべて配置することが不可能な場合, ラベルの大きさをスケールダウンして, すべてのラベルを配置することを考える. そのとき, ラベルの大きさをできる限り大きくする. この問題は, ラベルサイズ最大化問題と呼ばれる. 本研究で考える問題に照らし合わせて考えると, 障害物に対応するピクセルが重み $-\infty$ を持つピクセル平面が与えられているとみなせる. そのときに, ピクセル平面のグリッド上に与えられた点を長方形のどこかの隅に持つように長方形を配置して, 長方形の重みの最小値を最大化する問題とみなせる. 一般的なラベルサイズ最大化問題は NP 困難であることが, Formann と Wagner[4] により示されている. したがって, グリッド上に点が与えられない場合は, さらに難しい問題であると予想される. 以上のことから, 多項式時間で解くことはほぼ不可能であることが予想される.

どちらの問題も厳密に解き, 最適解を求めるには膨大な時間が必要になってしまう. そこで, この問題に対して新たな入力を加えることにより, この問題の難しさがどのようになるのかを解析する. もし, 効率的に解くことができるならば, この問題を用いてヒューリスティックを設計することができると考えられる.

1.2 研究の目的

本論文では, 各ピクセルの重みを表す行列と, その平面のグリッド上に k 個の点が与えられたとき, 前述した最適化基準を用いて長方形やタブローを配置するアルゴリズムを提案する. ここで, 配置される長方形やタブローはグリッド上に与えられた点 p_i を左上, あるいは左下に持つ. これらの問題を効率的に解くことで, グリッド上に点が与えられない場合の問題を解くヒューリスティックを設計できると考えられる. そして, これらの問題を表 1.1 の時間計算量で解けることを示す.

2 隅位置情報を持つ場合の長方形の最適配置問題

この節では, 入力として配置する長方形の隅位置情報が与えられる場合の最適配置問題について述べ

表 1.1. 時間計算量と空間計算量

	最小値最大化	総和最大化
長方形	$O(kn^2 + n^2 \log n)$	$O(k^{2k+1}n^2)$
タブロー	$O(n^2 \log \Gamma_T)$	$O(k^{2k}n^3)$

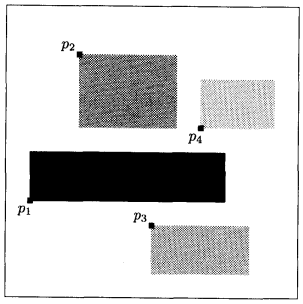

図 2.1. 隅位置情報を持つ長方形の配置の例

る. 隅位置情報とは, ピクセル平面のグリッド上に与えられる点で表される. ある 1 つの長方形はその点を必ず 2 つの隅 (左上, 左下) のうちのどこかで, その点を含むように配置される (図 2.1 参照). このような条件を追加した上で, 最小値最大化問題と総和最大化問題についてそれぞれ説明する.

2.1 最小値最大化問題

はじめに, 最小値最大化問題について述べる. この問題は重みの最小値が最大となるように長方形を配置する問題である. また, 長方形を配置するときは, それぞれの長方形が重ならないように配置する. 長方形配置の最小値最大化問題は次のように定式化される.

長方形配置の最小値最大化問題

入力 それぞれのピクセルが重みを持つ $n \times n$ のピクセル平面 P,
グリッド上に配置される k 個の点集合 $S = \{p_1, p_2, \ldots, p_k\}$,
長さが k のビット配列 $\mathbf{b} = b_1 b_2 \cdots b_k$

出力 長方形の重みの最小値が最大となる長方形配置

制約 配置される長方形は与えられた点を左下, あるいは左上に持つ.
それぞれの長方形は重ならない.

最小値最大化問題に対するアルゴリズムを説明する前に, この問題を解く上で重要となる長方形配置の判定問題について述べる. この問題は, あるしきい値 θ が与えられたとき, すべての長方形を θ 以上の重みで配置できるかどうかを判定する問題である.

判定問題を解くアルゴリズムは次のようになる.

アルゴリズム 2.1.
STEP1 与えられた点を x 座標でソートする.
STEP2 右にある点 (x 座標の大きい点) から長方形を配置する. そのとき, しきい値 θ 以上の重みを持つ長方形の中から, 高さが一番低い長方形を配置する. もし, θ 以上の重みを持つ長方形配置が存在しないならば, No を返し, アルゴリズムを終了する. k 個の長方形が配置できたならば, Yes を返してアルゴリズムを終了する.

配置される長方形の高さは一番低いものを選ぶので, 配置された長方形よりも高さが低く, かつ重みがしきい値以上となる配置は存在しないことがわかる. よって, この高さは重みがしきい値以上になるために必ず必要な長方形の高さとなる. そして, 他の点を含む長方形が配置可能な領域内で制約を満たせない場合は, 与えられたしきい値以上ではすべての長方形を配置することができない. したがって, このアルゴリズムは正しい判定を行うことがわかる.

このアルゴリズムの計算時間を解析する. **STEP1** では n 個の点をソートするので, $O(n \log n)$ 時間かかる. そして, **STEP2** では k 個の点がサイズ n^2 のピクセル平面を探索することになる. したがって, 計算時間は $O(kn^2)$ 時間となるが, それぞれのピクセルは高々1 回しか探索されないので, 全体の探索にかかる時間は $O(n^2)$ となる. よって, 長方形配置の判定問題は $O(n^2)$ 時間で解くことができる. また, この問題の空間計算量は入力されたピクセル平面を記憶する必要があるため, ピクセル平面のサイズと同じ $O(n^2)$ である.

次に, この判定問題を用いた最小値最大化問題を解くアルゴリズムについて述べる. 前処理として, 与えられたピクセル平面における長方形の重みの最小値と最大値を求める. ここで, 長方形の重みの絶対値を Γ_R とすると, 探索する要素数は $O(\Gamma_R)$ となる. これらの要素を解の候補として, 二分探索を行い, 最適値を求める. そのとき, 長方形配置の判定問題をサ

ブルーチンとして用いる．最小値最大化問題を解くアルゴリズムは次のようになる．

アルゴリズム 2.2.
STEP1 解の候補の中央値をしきい値 θ として長方形配置の判定問題を解く．
STEP2 判定問題の出力が Yes ならば，θ よりも小さい値を解の候補から削除する．No ならば，θ 以上の値を解の候補から削除する．
STEP3 解の候補が 1 つになるまで **STEP1** と **STEP2** を繰り返す．

STEP2 において，判定問題の出力が Yes ならば，θ よりも小さい値は最適解となり得ないので，解の候補から削除して良い．同様に No ならば，θ 以上の値では長方形の配置が存在しないことがわかるので，それらの値は最適値になり得ない．

これまでに述べてきたアルゴリズムの全体の計算時間を解析する．まず，二分探索は要素数が $O(\Gamma_R)$ の配列に対して行うので，$O(\log \Gamma_R)$ 時間となる．そして，1 回の探索で長方形配置の判定問題を 1 回解くことになるので，かかる時間は $O(n^2)$ である．したがって，全体の計算計算量は $O(n^2 \log \Gamma_R)$ となる．さらに，ランダム選択手法を用いることで，計算時間を $O(kn^2 + n^2 \log n)$ にすることができる．

2.2 総和最大化問題

ここでは，長方形の重みの和が最大となるように配置する総和最大化問題について述べる．前節と同様に，配置される長方形は互いに重なることはないものとする．長方形配置の総和最大化問題は次のように定式化される．

長方形配置の総和最大化問題

入力 それぞれのピクセルが重みを持つ $n \times n$ のピクセル平面 P，
グリッド上に配置される k 個の点集合 $S = \{p_1, p_2, \ldots, p_k\}$，
長さが k のビット配列 $\mathbf{b} = b_1 b_2 \cdots b_k$
出力 長方形の重みの総和が最大となる長方形配置
制約 配置される長方形は与えられた点を左下，あるいは左上に持つ．
それぞれの長方形は重ならない．

この問題を解くアルゴリズムは次のようになる．

アルゴリズム 2.3.
STEP1 入力されたグリッド点を通る水平線分と垂直線分をひき，ピクセル平面を分割する．
STEP2 配置される長方形の縦と横の長さを推測する．
STEP3 **STEP2** での推測をもとに有向グラフを作成する．
STEP4 **STEP3** で作成した有向グラフを用いて重み和を計算する．
STEP5 **STEP2**〜**STEP4** をすべての推測について実行し，重み和の最大値を最適解として出力する．

次にそれぞれのステップについて具体的な手順を述べる．そのときに時間計算量と空間計算量についても議論する．

STEP1 与えられたグリッド点を通る水平線分と垂直線分をひき，ピクセル平面を分割する．このとき，ピクセル平面は小さい長方形に分割され，本論文ではこの長方形をセルと呼ぶ．このセルは $O(k^2)$ 個存在する．

STEP2 このステップでは長方形がどこのセルまで入り込むのかを推測する．したがって，1 つの点に対して推測の候補は $O(k^2)$ 存在するので，全体で $O(k^{2k})$ 存在する．各長方形の縦と横を決めた後，その推測に対して，ラベルをセルに付ける．ラベルは右にある点から割り当てていき，点 p_i を含む隅位置に持つ長方形が入り込むことができるセルにラベル R_{p_i} をつける．ここで，ラベル R_{p_i} を持つセルの集合を $Rect_{p_i}$ とし，この集合の形に注目する．本論文では，ラベルの集合がすべて長方形の形になっている割り当てを正しい割り当てと定義し，このような割り当てに対して **STEP3** 以降の操作を実行する．

このようにセルにラベルをつけると，次の補題が成り立つ．

補題 2.4. 1 つのセルにつくラベルの数は高々2 つである．

STEP3 はじめに，与えられた点を通る垂直線分でピクセル平面を分割し，分割してできたそれぞれの部分平面をスラブと呼ぶ．ここで，左から j 列目に

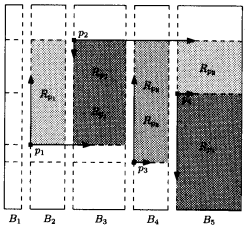

図 2.2. スラブの要素

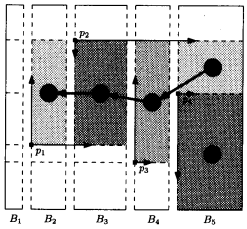

図 2.3. 有向グラフ

あるスラブを B_j とする. 次に, スラブごとに共通のラベルを持つセルを結合し, それをスラブの要素と呼ぶ. スラブ B_j にあり, その要素がラベル R_1, R_2 を持つならば, $Z_{B_j}(R_1, R_2)$ と表現する. 要素につけられるラベルは, 結合前のそれぞれのセルが持っているラベル集合の和集合とする.

次に, 作成したスラブの要素を点集合とし, 要素間の関係を辺集合とする有向グラフを作る. スラブ B_j にある要素 Z_{B_j} とスラブ B_{j+1} にある要素 $Z_{B_{j+1}}$ に対して, 共通のラベルが存在するならば, 要素 $Z_{B_{j+1}}$ から要素 Z_{B_j} へ有向辺をつける. 例えば, 図 2.2 の $Z_{B_4}(R_{p_2}, R_{p_3})$ と $Z_{B_5}(R_{p_2})$ は共通のラベル R_{p_2} を持っている. よって, 要素 $Z_{B_5}(R_{p_2})$ から要素 $Z_{B_4}(R_{p_2}, R_{p_3})$ への有向辺をつける. この操作をすべての要素に対して実行と図 2.3 のようなグラフができる. このステップではすべてのセルを高々1 回調べて, 要素を作るので $O(k^2)$ 時間かかる. また, グラフの作成ではスラブごとにセルを高々1 回調べるので, 先程と同じ時間がかかる. よって, 全体では $O(k^2)$ 時間となる.

このステップで作成される有向グラフに対して, 次の補題が成り立つ.

補題 2.5. 正しい割り当てならば, 有向グラフはパスの集合になる.

STEP4 動的計画法を用いて, **STEP3** で得られた有向グラフにおける最適値を求める. 動的計画法の計算はある有向グラフに対して, $O(kn^2)$ 時間で行うことができる.

STEP5 推測の候補数が $O(k^{2k})$ あり, **STEP3** と **STEP4** にかかる計算時間の合計は $O(kn^2)$ であるから, アルゴリズム 2.3 にかかる時間計算量は $O(k^{2k+1}n^2)$ となる. また, 空間計算量は **STEP4** と同じになるので, $O(kn^2)$ となる.

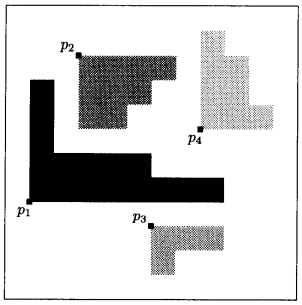

図 3.1. 隅位置情報を持つタブローの配置の例

3 隅位置情報を持つ場合のタブローの最適配置問題

この節では前節と同様の入力に対して, 図 3.1 のようにタブローを最適に配置する問題について述べる. タブローとは点 p を同じ頂点に持つ長方形の和集合領域で表される図形である. この点 p を隅位置として与え, 重みの最小値を最大化する問題と重みの和を最大化する問題について説明する.

3.1 最小値最大化問題

はじめに, 最小値最大化問題について述べる. この問題は配置されるタブローの重みの最小値を最大化する問題である. 条件は長方形の場合と同様に, タブローを重ならないように配置することである. タブロー配置の最小値最大化問題は次のように定式化される.

タブロー配置の最小値最大化問題

入力 それぞれのピクセルが重みを持つ $n \times n$ のピクセル平面 P,
グリッド上に配置される k 個の点集合 $S = \{p_1, p_2, \ldots, p_k\}$,
長さが k のビット配列 $\mathbf{b} = b_1 b_2 \cdots b_k$

出力 タブローの重みの最小値が最大となる長方形配置

制約 配置されるタブローは与えられた点を左下, あるいは左上に持つ.
それぞれのタブローは重ならない.

この問題を解くアルゴリズムは, 配置する図形が異なるだけで, 流れはアルゴリズム 2.2 と同じである.

したがって, 計算時間は探索する範囲のサイズと判定問題の計算時間に依存する.

はじめに, タブロー配置の判定問題は長方形の場合と同様の操作を行うことで解くことができる. そのときにかかる時間は, $O(n^2)$ 時間である. 次に最適値の探索は, タブローの重みが最小となる値と最大となる値の間にあるすべての値に対して行われる. したがって, 探索の範囲は配置可能なすべてのタブローの中で重みの絶対値が最大となる値を Γ_T とすると, $O(\Gamma_T)$ となる. 以上のことから, 最小値最大化問題を解くのに $O(n^2 \log \Gamma_T)$ 時間かかる. 空間計算量は判定問題を解く際に必要な $O(n^2)$ と同じである.

次にランダム選択による手法について考える. 長方形の場合, 配置可能な候補数は 1 つの点に対して $O(n^2)$ であるから, k 個の点では $O(kn^2)$ である. よって, 配置可能なすべての長方形の重みを用いてランダム選択を行うことができた. タブローの場合では, 配置可能な候補数が $\binom{2n}{n} \leq 2^{2n}$ であるから, すべての場合について重みを求めるのには膨大な時間計算量と空間計算量が必要になる. したがって, タブロー配置においてランダム選択を用いるのは現実的ではないことがわかる.

3.2 総和最大化問題

ここでは, 長方形の重みの和が最大となるようにタブローを配置する問題について述べる. タブロー配置の総和最大化問題は次のように定式化される.

タブロー配置の総和最大化問題

入力 それぞれのピクセルが重みを持つ $n \times n$ のピクセル平面 P,
グリッド上に配置される k 個の点集合 $S = \{p_1, p_2, \ldots, p_k\}$,
長さが k のビット配列 $\mathbf{b} = b_1 b_2 \cdots b_k$

出力 タブローの重みの総和が最大となるタブロー配置

制約 配置されるタブローは与えられた点を左下, あるいは左上に持つ.
それぞれのタブローは重ならない.

この問題も最小値最大化問題の場合と同様に, 流れはアルゴリズム 2.3 と同じである. 計算時間は, **STEP4** における最適解の計算がタブローの場合, 1 つの有向グラフに対して $O(n^3)$ 時間かかる. よって, 全体の計算時間は $O(k^{2k}n^2)$ となる. また, 空間計算量は動的計画法を解くのに $O(n^3)$ かかるため, 全体では $O(n^3)$ となる.

4 まとめ・今後の課題

本論文では隅位置情報が与えられる場合の長方形とタブローの最適配置問題を解くアルゴリズムを提案した. 特に長方形の場合, 隅位置情報を入力として与えないとき, 最小値最大化問題も総和最大化問題も共に NP 困難であると知られている. この 2 つの問題に対して, 隅位置情報を与えることにより, 最小値最大化問題は多項式時間で解くことができ, 総和最大化問題は FPT であることを示した.

参考文献

[1] T. Asano, D. Z. Chen, N. Katoh, and T. Tokuyama. Efficient algorithms for optimization-based image segmentation. *Int. J. Comput. Geometry Appl.*, 11(2):145–166, 2001.

[2] S. E. Bae and T. Takaoka. Algorithm for disjoint maximum subarrays. In *International Conference on Computational Science (1)*, pages 595–602, 2006.

[3] J. Chun, R. Kasai, M. Korman, and T. Tokuyama. Algorithms for computing the maximum weight region decomposable into elementary shapes. In *ISAAC*, pages 1166–1174, 2009.

[4] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Symposium on Computational Geometry*, pages 281–288, 1991.

[5] M. Korman. *Theory and Applications of Geometrix Optimization Problems in Rectilinear Metrix Spaces.* PhD thesis, Tohoku University, 2009.